C'est quoi un microcontrôleur ?

Un microcontrôleur est un circuit intégré programmable il comporte tous les éléments d'une structure à base de microprocesseur il comporte principalement :

- Microprocesseur
- Mémoire de données RAM (« volatile » données perdue en cas de coupure de l'alimentation)
- Mémoire programme ROM « non volatile » différent types (ROM, PROM, EPROM, EEPROM ou FLASH)
- Des interfaces parallèles répartir sur plusieurs PORTS (maximum 8 bits) ces ports peuvent être configurées en entrée ou sortie.
- Des convertisseurs analogique numérique pour traiter des signaux analogiques .
- Des Timers pour réaliser des temporisations ou faire des comptages.

Est-ce qu'il ya une seule famille de microcontrôleur ?

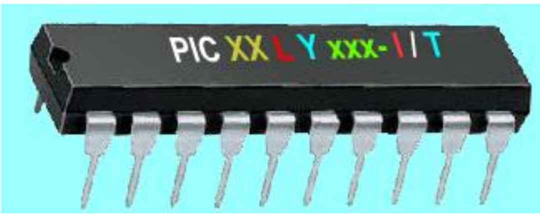
Non , plusieurs constructeurs se partagent le marché de microcontrôleurs citons :

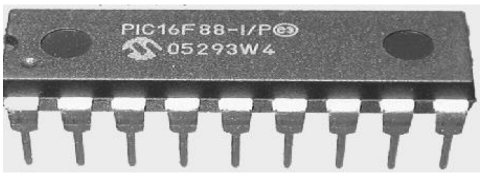
INTEL, **MOTOROLA**, **ATMEL**, **PHILIPS**, **MICROCHIP** avec ses **PIC** très populaires qui nous intéresse dans notre programme officiel.

Comment identifier un microcontrôleur de **MICROCHIP** ?

Un PIC est généralement identifié par une référence de la forme suivante :

PIC XX(L)Yxxx- I/T





XX : famille du composant, actuellement « **12, 14, 16, 17 et 18** ».
Actuellement les modèles Microchip, sont classés en 3 grandes familles, comportant chacune plusieurs références. Ces familles sont :

- **Base-line** : Les instructions sont codées sur **12 bits**.
- **Mid-line** : Les instructions sont codées sur **14 bits**.
- **High-end** : Les instructions sont codées sur **16 bits**.

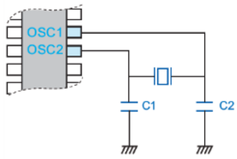
- **L** : tolérance plus importante de la plage de tension.
  - Sans **L** : Alimentation standard de 3 à 5,5V.
  - Avec **L** : Alimentation étendue de 2 à 5,5V.
- **Y** : type de mémoire programme :
  - **C**: EPROM ou EEPROM.
  - **CR**: PROM.
  - **F** : Flash.
- **xxx** : Identificateur. (référence du circuit)
- **I** : Gamme de température :
  - Sans I : de 0 à 70 °C.
  - Avec I : de -40 à 85°C.
- T : Type de boîtier. Ex « P » PDIP

Broches pour oscillateur

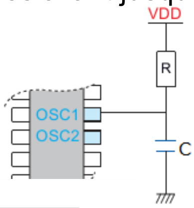
OSC1 et OSC2 ou CLKIN et CLKOUT

Pour qu'un microcontrôleur fonctionne correctement, il est nécessaire d'utiliser un signal d'horloge dont le rôle est de cadencer l'exécution des instructions du programme.

Les broches OSC1 et OSC2 permettent de raccorder un oscillateur externe au PIC pour fournir un signal d'horloge. On peut utiliser trois types d'oscillateurs externes:

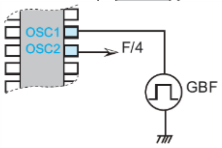


- **Un quartz** (XT ou CRYSTAL) On peut avoir de fréquences allant jusqu'à 20 MHz selon le type microcontrôleur;



- **Un oscillateur (RC)**

La fréquence de l'oscillation est fixée par VDD,R et C.



- **Horloge externe**

Une horloge externe au microcontrôleur comme GBF ou oscillateur à base de circuits intégrés.

Les fameux PIC utilisés dans nos applications sont :

Caractéristiques	16F84A	16F628A	16F876A	16F877A
Nombre de broches	18	18	28	40
Mémoire programme	1024(**1K**)	2048 (**2K**)	8192 (**8K**)	8192 (**8K**)
Oscillateur interne	non	oui	non	non
Entrées -sorties	13 2 PORTS PORTA: 5 / PORTB: 8	16 2 PORTS PORTA: 8 / PORTB: 8	22 3 PORTS PORTA: 6 / PORTB: 8 / PORTC: 8	33 5 PORTS PORTA: 6 / PORTB: 8 / PORTC: 8 / PORD: 8 / PORTE: 3
Timers	TMR0 (8bits)	TMR0 (8bits) TMR1 (16bits) TMR2 (8bits)	TMR0 (8bits) TMR1 (16bits) TMR2 (8bits	TMR0 (8bits) TMR1 (16bits) TMR2 (8bits
Comparateurs analogiques	non	2	2	2
Convertisseur analogique	**non**	**non**	**1CAN 10 bits 5canaux** AN0: RA0 / AN1: RA1 / AN2: RA2 / AN3: RA3 / AN4: RA5	**1CAN 10 bits 8canaux** AN0: RA0 / AN1: RA1 / AN2: RA2 / AN3: RA3 / AN4: RA5 / AN5: RE0 / AN6: RE1 / AN7: RE2
CC**P** (**PWM** ou **MLI**)	**non**	**oui**	**oui**	**oui**

Différents registres utilisés

Registre									Après reset
TRISx	Registre de direction pour configurer les entrées et les sorties **1 : entrée** **0 : sortie**								11111111
OPTION_REG	Registre de configuration du TIMER0								
	$\overline{RBPU}$	**INTEDG**	**TOCS**	**TOSE**	**PSA**	**PS2**	**PS1**	**PSO**	11111111
INTCON	Registre de configuration des interruptions								
	GIE	**EEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**	0000000x
ADCON1	Registre de configuration du convertisseur								
	ADFM	**-**	**-**	**-**	**PCFG3**	**PCFG2**	**PCFG1**	**PCFG0**	0---0000
ADRESL **ADRESH**	Le convertisseur C.A.N fournit un nombre binaire naturel de 10 bits (B9 B8 B7 B6 B5 B4 B3 B2 B1 B0). 2 registres(2 X 8 bits) sont nécessaire pour stocker le résultat de la conversion. Ce sont les registres :ADRESL et ADRESH								
CMCON	Pour le circuit 16F628A pour désactiver les comparateurs analogiques et rendre le PORTA numérique **CMCON=7**								

LE TIMER TMR0

Le registre TMR0 est un compteur programmable de 8 bits (de 0 à 255).

La configuration du TMR0 est assurée par le registre OPTION **« OPTION_REG »**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RBPU	**INTEDG**	**TOCS**	**TOSE**	**PSA**	**PS2**	**PS1**	**PSO**

PS2	PS1	PSO	Diviseur
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

Le choix de l'horloge se fait à laide du **bit 5** du registre **OPTION_REG** **« TOCS »**

- **TOCS = 0** Horloge interne « **mode Temporisateur** »
- **TOCS = 1** Horloge externe « **mode COMPTEUR**»

Dans le cas de l'horloge externe, le **bit 4 « TOSE »** du registre **OPTION_REG** permet de choisir le **front** sur lequel le **TIMER0** s'incrémente :

- **TOSE = 0** incrémentation **sur fronts montants**
- **TOSE = 1** incrémentation **sur fronts descendants**

Quelque soit l'horloge choisie, on peut la faire passer dans un diviseur de fréquence programmable (prescaler) dont le rapport est fixé par les bits **PS0,PS1 et PS2** du registre **OPTION_REG** « voir tableau »

L'affectation ou non du prédiviseur se fait à l'aide du **bit 3 « PSA »** du registre **OPTION_REG**

- **PSA =0** on utilise le prédiviseur.
- **PSA =1** pas de prédiviseur.

Bit 6 :INTEDG « INTerrupt Edge » : dans le cas où on utilise l' interruption externe avec RB0

- Si **INTEDG** = 1, on a interruption si le niveau sur RB0 passe de 0 vers 1. « front montant »
- Si **INTEDG** = 0, l'interruption s'effectuera lors de la transition de 1 vers 0. « front descendant »

Bit 7 : RBPU: Quand ce bit est mis à 0, une résistance de rappel au +5 volt est placée sur chaque broche du PORTB

NOTION D'INTERRUPTION

C'est quoi une interruption ?

Une interruption est un événement qui provoque l'arrêt d'un programme en cours d'exécution pour aller exécuter un autre programme appelé programme d'interruption (ou routine).

À la fin du programme d'interruption, le microcontrôleur reprend le programme principal à l'endroit où il s'est arrêté.

Le registre **INTCON** (INTerrupt CONtroller) est le registre principal de contrôle et de gestion des interruptions. Registre **INTCON** **pour PIC16F84A**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	**EEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**

Registre INTCON

On s'intéresse uniquement au deux types d'interruptions

- À l'interruption sur la broche RB0
- À l'interruption RB4 à RB7

- RB0/INT : Une interruption peut être générée lorsque, la broche RB0, encore appelée INTerrupt pin, étant configurée en entrée, le niveau qui lui est appliqué est modifié.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	**EEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**
1	**0**	**0**	**1**	**0**	**0**	**0**	**0**

Registre INTCON = 90$_{(Hex)}$

- PORTB : De la même manière, une interruption peut être générée lors du changement d'un niveau sur une des pins RB4 à RB7.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	**EEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**
1	**0**	**0**	**0**	**1**	**0**	**0**	**0**

Registre INTCON = 88$_{(Hex)}$

N.B :Lorsque le mécanisme de l'interruption RB4 à RB7 se déclenche le microcontrôleur verrouille le bit indicateur RBIF à 1, une opération de lecture sur le port B est nécessaire pour déverrouiller l'accès au bit RBIF afin de pouvoir le remettre à 0.

CONVERTISSEUR

Les microcontrôleurs PIC 16F876 et 16F877 possédèrent un convertisseur analogique numérique sur 10 bits ,ce dernier permet de convertir une tension analogique comprise entre Vref- et Vref+ en une valeur numérique comprise entre 0 et 1023 .

Pour exploiter ce convertisseur il est nécessaire de configurer certains registres dans le microcontrôleur, Dans notre cas on intéresse au registre **ADCON1**

Registre ADCON1 :

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
bit 7	bit 6			bit 3	bit 2	bit 1	bit 0

4 bits PCFG				PIC 16F877								Tensions De références	
							PIC 16F876						
				PORTE			PORTA						
PCFG3	**PCFG2**	**PCFG1**	**PCFG0**	**AN7/RE2**	**AN6/RE1**	**AN5/RE0**	**AN4/RA5**	**AN3/RA3**	**AN2/RA2**	**AN1/RA1**	**AN0/RA0**	**Vref+**	**Vref-**
0	0	0	0	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}
0	0	0	1	A	A	A	A	Vref+	A	A	A	RA3	V_{SS}
0	0	1	0	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}
0	0	1	1	D	D	D	A	Vref+	A	A	A	RA3	V_{SS}
0	1	0	0	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}
0	1	0	1	D	D	D	D	Vref+	D	A	A	RA3	V_{SS}
0	1	1	X	D	D	D	D	D	D	D	D	V_{DD}	V_{SS}
1	0	0	0	A	A	A	A	Vref+	Vref-	A	A	RA3	RA2
1	0	0	1	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}
1	0	1	0	D	D	A	A	Vref+	A	A	A	RA3	V_{SS}
1	0	1	1	D	D	A	A	Vref+	Vref-	A	A	RA3	RA2
1	1	0	0	D	D	D	A	Vref+	Vref-	A	A	RA3	RA2
1	1	0	1	D	D	D	D	Vref+	Vref-	A	A	RA3	RA2
1	1	1	0	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}
1	1	1	1	D	D	D	D	Vref+	Vref-	D	A	RA3	RA2

A : entrée analogique D : entrée numérique **V_{DD} =V_{CC}= 5V ; V_{SS} = GND = 0 V**

Pour récupérer les **5 bits du port A** et **3 bits du port E** en tant que **I/O numériques (digitales)** il faut **ADCON1 =06$_{(H)}$**

Les bases du compilateur Mikropascal :

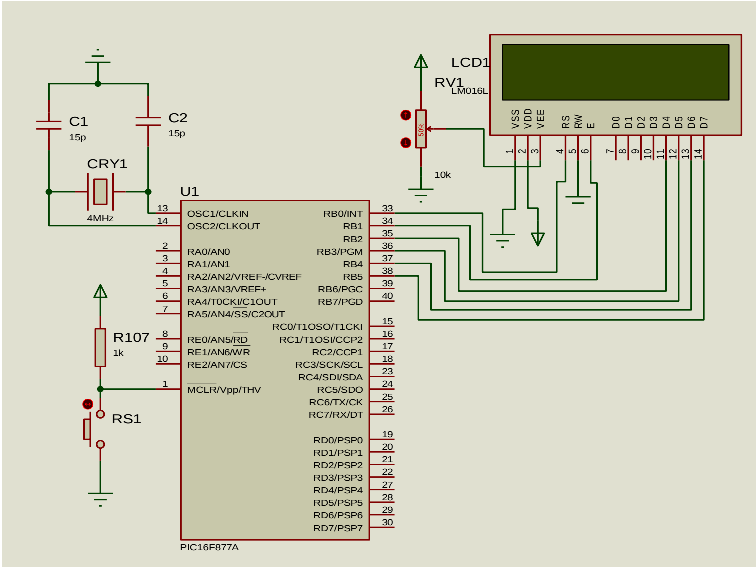
Le décimal	A = 12
L'hexadécimal	A = $0C ou A = 0x0C
Le binaire	A = %0001100

Instructions spécifique au compilateur Mikropascal pro

MODULE LCD

Instructions spécifique au compilateur Mikropascal pro pour l'afficheur LCD

Les variables suivantes doivent être définies dans tous les projets utilisant la bibliothèque d'affichage à cristaux liquides LCD:




```
// Connections du module LCD
var LCD_RS : sbit at RB0_bit;
var LCD_EN : sbit at RB1_bit;
var LCD_D4 : sbit at RB2_bit;
var LCD_D5 : sbit at RB3_bit;
var LCD_D6 : sbit at RB4_bit;
var LCD_D7 : sbit at RB5_bit;

var LCD_RS_Direction : sbit at TRISB0_bit;
var LCD_EN_Direction : sbit at TRISB1_bit;
var LCD_D4_Direction : sbit at TRISB2_bit;
var LCD_D5_Direction : sbit at TRISB3_bit;
var LCD_D6_Direction : sbit at TRISB4_bit;
var LCD_D7_Direction : sbit at TRISB5_bit;
// FIN
```

Lcd_Init () ; **//** Initialisation de l'LCD

Lcd_Out(1, 2, 'BRAVO') ; // écrire BRAVO sur l'LCD à partir de la ligne 1 ,colonne 2

Lcd_Chr(2, 3, i); // écrire la caractère équivalent en code ASCII i sur l'LCD à partir de la ligne 2 ,colonne 3

Lcd_Cmd exemples :

Lcd_Cmd(_LCD_CLEAR); // effacer l'LCD

Lcd_Cmd(_LCD_CURSOR_OFF) ; // supprimer le curseur de L'LCD

Lcd_Cmd(_LCD_FIRST_ROW) // **Déplacer le curseur à la 1ère ligne**

Lcd_Cmd(_LCD_SECOND_ROW) **//Déplacer le curseur à la 2ème ligne**

GESTION D'UN CLAVIER

Le mikroPascal PRO pour PIC fournit une bibliothèque pour travailler avec des claviers « en bloc de touches 4x4 ». Les routines de bibliothèque peuvent également être employées avec le bloc de touches 4x1, 4x2, ou 4x3.

L'utilisation de ce clavier nécessite un port obligatoirement bidirectionnel 8 bits.



Instructions spécifique au compilateur « Mikropascal pro » pour le Clavier

Var keypadPort : byte at PORT… ; // Pour la connexion du clavier au PORT. .. considéré (8 bits)
Var Kp :byte ; // on définit une variable de type octet

Keypad _Init(); // Initialisation du clavier

Kp:=Keypad_key_Press(); // lecture de code de la touche « touche enfoncée » de 1 à 16.

Kp:=Keypad_key_click(); // lecture de code de la touche « touche enfoncée puis libérée »de 1 à 16.

MODULE CONVERTISSEUR (C.A.N)

Instructions spécifique au compilateur Mikropascal pro pour le module conversion

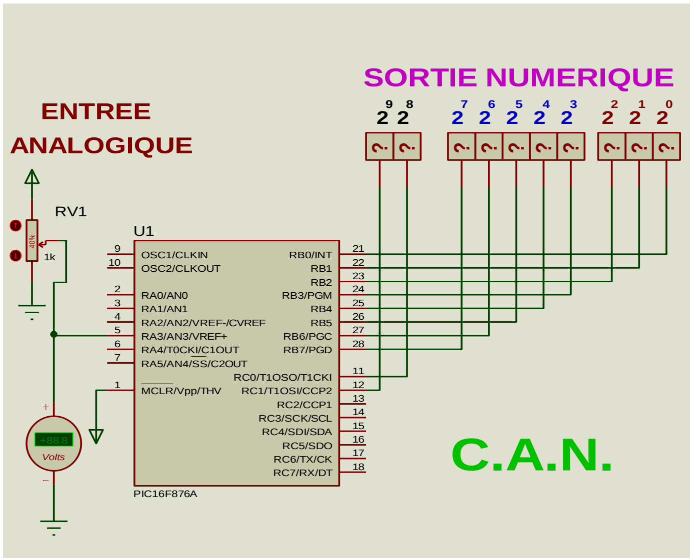
ADC_Init() ; // Initialise le module convertisseur et le configurer avec les réglages suivants:

Vref-=0 ;Vref+ = 5V , Utilisation de l'horloge interne pour la conversion.

N : word // déclaration d'une variable de type word

N := ADC_Get_Sample(3) // lecture de la valeur lue par le convertisseur sur le canal 3

N := ADC_Read (3) // lecture après initialisation et démarrage de la conversion sur le canal 3



LE MODULE M.L.I (PWM): PULSE WIDTH MODULATION

Instructions spécifique au compilateur Mikropascal pro pour le module PWM

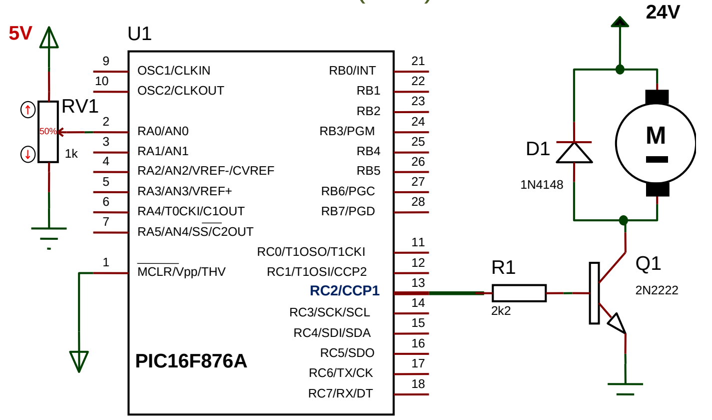
PWM1_Init(1000) // Initialise le module PWM de la sortie CCP1 (RC2) à la fréquence 1000Hz:

PWM1_start() // Démarrage du module PWM et sortie du signal sur la broche CCP1(RC2)

PWM1_Set_duty(N) // Change le rapport cyclique α du signal sortant sur la broche CCP1 (RC2)

avec N variant de 0 à 255 $\alpha = \frac{N}{255}$

PWM1_stop // Arrêter le module PWM de la sortie CCP1 (RC2)



Librairie de Conversions

ByteToStr **(procédure)** ByteToStr(input : byte; **var** output : **array**[3] **of** char); **Exemple** **var** t : byte; txt : **array**[3] **of** char; ... t := 24; **ByteToStr(t, txt);** *// txt is " 24" (ici il y a un espace)*	**ShortToStr** **(procédure)** ShortToStr (entree : short; **var** sortie: **array**[4] **of** char); **Exemple** **var** t : short; txt : **array**[4] **of** char; ... t := -24; ShortToStr(t, txt); *// txt est " -24" (ici il y a un espace)*

wordToStr **(procédure)**
 	
wordToStr(entree : word; **var** sortie : **array**[5] **of** char);
 	
Exemple
 	
var t : word;
 	
txt : **array**[5] **of** char;
 	
...
 	
t := 437;
 	
WordToStr(t, txt); *// txt est " 437" (ici il y a deux espaces)*
	intToStr **(procédure)**
	
	IntToStr(entree : integer; **var** sortie: **array**[6] **of** char);
	
	Exemple
	
	var entree : integer;
	
	txt : **string**[6];
	
	entree := -4220;
	
	IntToStr(entree, txt); *// txt est ' -4220' (ici il y a un espace)*